Informe final del proyecto

Gestión de Proyecto Software

2015-16

Cristina Lahoz Egea (544393)

Patricia Lázaro Tello (554309)

Jorge Martínez Lascorz (571735)

Alejandro Royo Amondarain (560285)

Jaime Ruiz-Borau Vizárraga (546751)

ÍNDICE

1 INTRODUCCIÓN

El producto a desarrollar consiste en una aplicación de recetas de comida para smartphones, concretamente dispositivos con sistema operativo Android, que ayude a personas sin grandes conocimientos de cocina a hacer la comida.

El equipo que desarrolla este proyecto se compone de cinco integrantes. Los cinco miembros del equipo componen el *equipo de desarrollo*, Cristina Lahoz es la *dueña del producto* y Alejandro Royo es el *Scrum Master*.

En las siguientes secciones del documento se explica el estado del producto al finalizar los dos sprints, y las diversas estrategias seguidas en el proceso durante la ejecución de ambos sprints. Las tres secciones que vienen a continuación se encuentran divididas en dos partes, una para el primer sprint y otra para el segundo sprint.

En la sección del *Producto* se plantea el estado actual del producto, incluyendo la pila de producto, plan de producto y planificación de lanzamientos, así como su diseño arquitectural.

En la sección del *Proceso* se explican las diversas estrategias de control de versiones, test y construcción automática de software seguidas, así como diagramas de burnup, velocidad del equipo y esfuerzos, y los resultados de la retrospectiva del sprint.

En la última sección, *Conclusiones*, se enumera el grado de cumplimiento de los objetivos de la evaluación del proyecto, así como un pequeño sumario del documento.

2 PRODUCTO

2.1 PLAN DE PRODUCTO

Nuestra visión

Fooding es una aplicación para Android que conciencia a las personas para que coman sano mediante un sistema de puntuación y rankings globales. Para comer sano no hay nada mejor que el hábito, y Fooding trata de crear precisamente esto, hábitos.

Sabiendo que no todos los usuarios tienen grandes conocimientos de cocina, se pone a su disposición una gran colección de recetas con instrucciones detalladas y paso a paso y diversas ayudas, como generación de listas de la compra o búsqueda de supermercados cercanos.

La aplicación está dirigida a estudiantes en pisos de estudiantes y personas jóvenes que viven solas o que no tienen tiempo para comer, pues son las que peor comen, ya sea por dejadez, falta de tiempo o de ganas.

Con Fooding, será muy fácil mantener buenos hábitos de comida, respetando el tiempo mínimo para comer, sin saltarse comidas y sin recurrir a la siempre presente comida basura.

Requisitos de la aplicación

La aplicación permitirá al usuario encontrar las recetas en función de los ingredientes que tenga disponibles en un momento determinado (tres ingredientes como máximo). La búsqueda de dichas recetas se podrá filtrar además mediante nombres, ingredientes a utilizar y por tipo de receta (carne, pescado, verdura, postre y pasta). Si el usuario no tiene algún ingrediente para completar una receta la aplicación buscará el supermercado más cercano y permitirá realizar la compra digitalmente. Para que el usuario pueda mantener un registro de las últimas comidas y repetir fácilmente la aplicación permitirá guardar las recetas favoritas.

Por otro lado, la aplicación fomenta la interacción con otros usuarios utilizando valoraciones sobre recetas y clasificaciones en función de dichas valoraciones. Además el usuario podrá competir con la comunidad compartiendo sus recetas y ganando puntos por ello. Estos puntos permitirán al usuario subir de nivel y ser más visible dentro de la comunidad. Así, cuando publique alguna receta esta

tendrá prioridad respecto al resto de publicaciones. Los puntos se podrán obtener contribuyendo a la comunidad de manera activa, esto es, compartiendo nuevas recetas de nuevos tipos de comida, valorar recetas de otros usuarios o recibir buenas valoraciones de las recetas ya publicadas.

Hoja de ruta

A continuación se describe la hoja de ruta a seguir durante el primer año de desarrollo:

El *primer lanzamiento* corresponde a enero de 2016: incluirá las funcionalidades correspondientes al buscador de recetas, sus diferentes filtros, localización del usuario y supermercados cercanos mediante GPS.

El *segundo lanzamiento* se realizará en abril de 2016: incluirá la primera parte de la comunidad de usuarios, con sistema de puntuación, valoración de recetas, ranking por usuarios y aportaciones por parte de los usuarios.

Una vez conseguida una aplicación una aplicación con una funcionalidad destacable, se irán realizando lanzamientos prácticamente mensuales/semanales, añadiendo las nuevas funcionalidades que se vayan implementando.

2.2 ANÁLISIS DE RIESGOS

2.2.1 Análisis de riesgos del primer sprint

En cuanto al análisis de riesgos que se llevó a cabo antes de la planificación del primer sprint, la aproximación inicial que se realizó, fue la siguiente:

En un principio se identificaron cuatro riesgos.

1. La Base de Datos o el servidor no responden
2. Algún miembro del equipo se ausenta en cualquier reunión.
3. Posibles fallos en la comunicación entre los miembros del equipo.
4. Mala relación entre los miembros del equipo.

De estos cuatro posibles riesgos el más importante y que por tanto, más se ha notado en el desarrollo de este primer sprint ha sido el tres, posibles fallos en la comunicación entre los miembros del equipo, ya que muchas de las reuniones o decisiones tomadas durante ese tiempo se han realizado mediante la aplicación Whatsapp, por lo que, esto ha supuesto un gran problema de perdida de información.

Para solucionar esto se ha decidido utilizar Whatsapp para alguna consulta puntual y para los temas más importantes se realizarán reuniones en las que estarán presentes todos los miembros del equipo.

El resto de riesgos presentados anteriormente se han mitigado correctamente con las estrategias de mitigación expuestas en el documento de la fase previa al primer sprint.

2.2.2 Análisis de riesgos segundo sprint

En el segundo sprint, se llevó a cabo un análisis más exhaustivo de los posibles riesgos y su mitigación correspondiente, quedando como resultado la siguiente lista de riesgos y mitigaciones:

Riesgo: Las herramientas de software utilizadas por el equipo en la construcción del producto no se encuentran integradas entre sí.

Probabilidad de que se produzca: 2/5

Impacto si se produce: 5/5

Mitigación: Descubrir posibles fallos de integración en las herramientas de software mediante un diagnóstico temprano y plantear soluciones lo antes posible.

Riesgo: El producto termina en fracaso para el cliente.

Probabilidad de que se produzca: 3/5

Impacto si se produce: 4/5

Mitigación: Comunicación frecuente con el cliente para identificar a tiempo los posibles fallos y corregirlos antes de la fecha de entrega.

Riesgo: Plazos de entrega rígidos; no se puede retrasar la entrega del producto de la fecha establecida.

Probabilidad de que se produzca: 3/5

Impacto si se produce: 4/5

Mitigación: Establecer plazos de entrega internos antes de la fecha de entrega establecida, e intentar cumplirlos en la medida de lo posible.

Riesgo: Los miembros del equipo no conocen o manejan con soltura las herramientas de software utilizadas en el desarrollo del producto

Probabilidad de que se produzca: 4/5

Impacto si se produce: 3/5

Mitigación: Formar a los miembros del equipo que no conozcan estas herramientas lo antes posible, y si es necesario, escribir tutoriales para ayudar a la formación.

Riesgo: Algunos (o todos) los miembros del equipo trabajan en múltiples proyectos y pueden no estar disponibles para el desarrollo del producto en algún momento.

Probabilidad de que se produzca: 5/5

Impacto si se produce: 3/5

Mitigación: Los miembros del equipo conocen la disponibilidad del resto de los integrantes y se planifican las entradas de la pila a realizar durante el sprint de acuerdo a la disponibilidad de los mismos.

2.3 PILA DEL PRODUCTO

2.3.1 Pila del producto primer sprint

A continuación se muestran las entradas de la pila que forman la pila del producto en el primer sprint:

Meter recetas (S)
Configuración Inicial (M)
Listado inicial de recetas (M)
Búsqueda por nombre (S)
Búsqueda por tipo (S)
Búsqueda por ingredientes (S)
Registro Usuarios (M)
Login de usuarios (S)
Logout de usuarios (S)
Verificar registro (S)
Login automático (S)

- **Meter recetas:**

  1. Existe una base de datos con la estructura necesaria para almacenar las recetas que mostrará la aplicación.

  2. Dicha base de datos contendrá almacenadas suficientes estructuras para que la aplicación sea utilizable. Dicha medida de "utilizable" vendrá dada por el dueño del producto.

- **Configuración inicial:**

  1. Existe un sistema que permite gestionar todas las solicitudes de la aplicación a la base de datos (No es necesario que esté configurado, esta entrada de la pila se refiere a montar el sistema para su uso/configuración más tarde).

- **Listado inicial de recetas:**

  1. Debe mostrar todas las recetas al usuario en una columna al iniciar la aplicación. La información mostrada para cada receta será una imagen que represente el tipo (verdura, carne, pescado) y el nombre de la receta.

- **Búsqueda por nombre:**

  1. Existe un cuadro de búsqueda donde se puede introducir texto y un botón de búsqueda.

  2. Al introducir texto y pulsar el botón de búsqueda, se listen aquellas recetas que contengan ese texto en el nombre.

  3. Los resultados obtenidos se mostrarán ordenados de forma ascendente o descendente en función de la elección del usuario.

- **Búsqueda por tipo:**

  1. Existe un cuadro donde se puede elegir el tipo de receta sobre el que se desea filtrar.

  2. Al elegir el tipo, se listen aquellas recetas que sean del susodicho tipo.

- **Búsqueda por ingredientes:**

  1. Existe un cuadro donde se pueden elegir los ingredientes por los que filtrar las recetas.

  2. Al elegir los ingredientes concretos, se listen aquellas recetas que contengan dichos ingredientes.

- **Registro usuarios:**

  1. Existen tres cuadros de texto donde se puedan introducir el nombre, el email y la contraseña del usuario a registrar, además de un botón para finalizar el registro.

- **Login usuarios:**

  1. Existen dos cuadros de texto donde se puedan introducir el email del usuario y su contraseña, además de un botón de login para validar el inicio de sesión.

- **Logout usuarios:**

  1. En el menú de inicio cuando haya una sesión abierta de un usuario se mostrará un botón para cerrar dicha sesión y la aplicación volverá al modo inicial de esta.

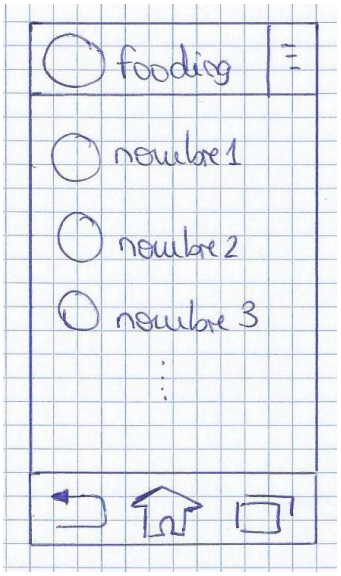
- **Verificar registro:**

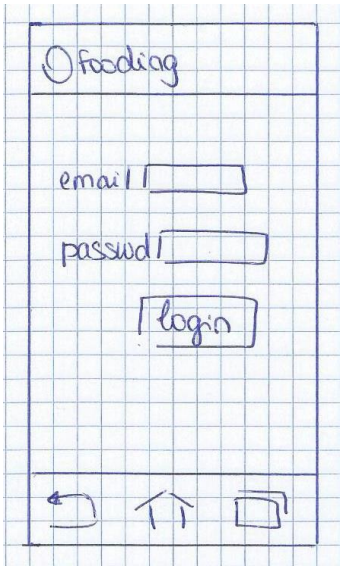
  1. Cuando un usuario se registra y pulsa el botón de finalizar se le enviará un correo electrónico a la cuenta de correo indicada con un enlace, en el que tendrá que pulsar para verificar que es la persona que solicita el registro.

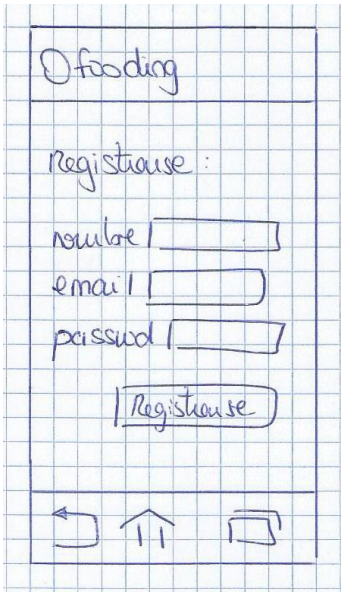
- **Login automático:**

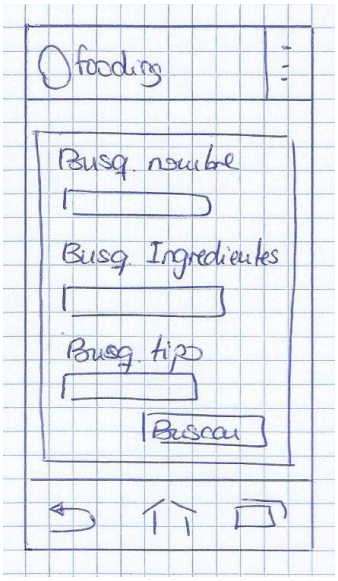
  1. El usuario cuando ya se haya registrado o iniciado sesión anteriormente a no ser que pulse el botón de logout permanecerá con su cuenta iniciada aunque cierre la aplicación.

A continuación se muestran una serie de prototipos iniciales de las pantallas:









Estos son algunos de los primeros diseños de la aplicación.

2.3.2 Pila del producto segundo sprint

Modificar comensales de las recetas
Crear y publicar recetas
Valorar recetas
Visualizar valoración de las recetas
Sistema de favoritos
Sistema de puntuación de usuarios
Pantalla de usuario registrado
Listado de usuarios
Mejora de la aplicación

- **Modificar comensales de las recetas:**
  1. Existen dos botones en la GUI que permiten modificar el número de comensales de una receta, aumentándolos o disminuyéndolos.

  2. El número mínimo de comensales es 1.

  3. La cantidad de cada ingrediente de la receta será la cantidad correspondiente a 1 comensal multiplicada por el número de comensales introducidos por el usuario.

- **Crear y publicar recetas:**
  1. Existe un botón en la pantalla del listado de recetas que redirige al usuario a una pantalla de creación de recetas.

  2. La pantalla de creación de recetas tiene un campo de texto para el nombre de la receta, un desplegable para el tipo de la receta, y una estructura para elegir hasta 3 ingredientes. Esta estructura consta de 3 campos diferentes, uno para la cantidad, donde solo se pueden poner números, otro para las unidades y un desplegable para el nombre del ingrediente. En la pantalla también hay un campo de texto para la elaboración de la receta y un botón para confirmar la información

introducida y un botón para confirmar la información introducida y crear la receta.

3. Las recetas tiene un campo, autor, que cuando se crea una receta se rellena automáticamente con la información del usuario logueado en ese momento.

4. Después de crear una receta, el usuario es redirigido a la pantalla de listado de recetas, con la lista de recetas actualizada.

5. Solamente lo usuarios registrados pueden crear y publicar recetas.

- **Valorar recetas**

  1. Existen dos botones en la pantalla de información de una receta que permiten a un usuario realizar una valoración sobre la receta que está visualizando. Uno de los botones, "Me gusta", aumenta el número de votos positivos en una unidad y el otro, "No me gusta", aumenta el número de votos negativos en una unidad.

  2. Cada usuario solamente puede valorar la receta con uno de los botones y una sola vez, aunque podrá cambiar su voto una vez realizado.

  3. Solamente los usuarios registrados pueden valorar una receta.

- **Visualizar valoración media de las recetas**

  1. Existe un apartado en la pantalla de información de la receta que permite a un usuario ver el porcentaje de votos positivos en función de los votos totales emitidos a dicha receta. Valoración = votos positivos / (votos positivos + votos negativos)

- **Sistema de favoritos**

  1. Existe una opción en el menú lateral de la GUI que permite visualizar las recetas seleccionadas como favoritas del usuario logueado en ese momento. Esta funcionalidad solo está disponible para usuarios logueados, por lo que si no lo está esta opción no aparecerá en el menú.

  2. Existe una opción en el listado de recetas y en la pantalla de visualización de una receta que permita añadir a favoritos o quitar de la lista de favoritos dicha receta.

- **Sistema de puntuación de usuarios**

  1. El sistema mostrará la puntuación del usuario en la pantalla de visualización del perfil de un usuario, así como en el listado de todos los usuarios.

  2. La puntuación del usuario será igual a la suma de votos positivos de todas sus recetas dividido entre la suma de todos los votos de todas sus recetas.

- **Pantalla de usuario registrado**

  1. Existe una opción en el menú lateral, "Mi cuenta", que permite ver los datos del usuario logueado en una pantalla. En caso de que el usuario no este logueado esta opción no aparecerá.

  2. En la pantalla de usuario se mostrará el nombre y el correo del usuario logueado en ese momento.

- **Listado de usuarios:**

  1. Existe un botón en el menú lateral de la aplicación que redirige al usuario a una pantalla que permite visualizar la lista de usuarios que hay registrados en la aplicación.

  2. Existe una opción en la pantalla del listado de usuarios que permite buscar a los usuarios por el nombre. Cuando el usuario selecciona el botón de búsqueda aparece un "pop-up" con una caja de texto donde se puede introducir el nombre de un usuario y un botón para confirmar el texto y realizar la búsqueda, esto redirige al listado de usuarios donde se mostrarán aquellos cuyo nombre coincide de forma parcial o completa con el texto introducido.

  El "pop-up" de búsqueda también permitirá ordenar los usuarios por nombre (alfabéticamente) y por puntuación.

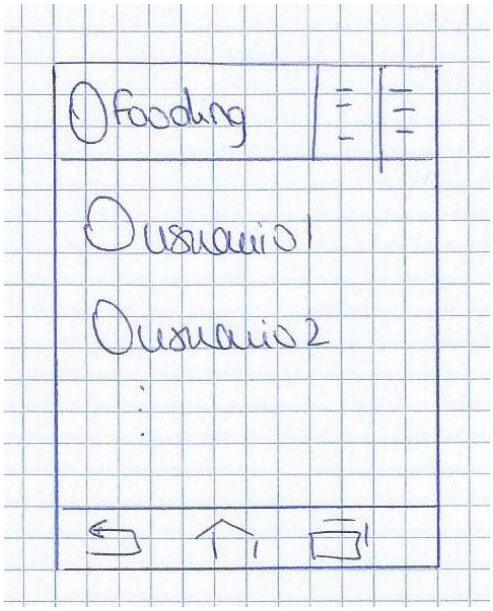
  3. Cuando se pulsa en un usuario se muestra un pop-up que muestra la información de ese usuario, el nombre, el listado de recetas introducidas por él y la puntuación del mismo.

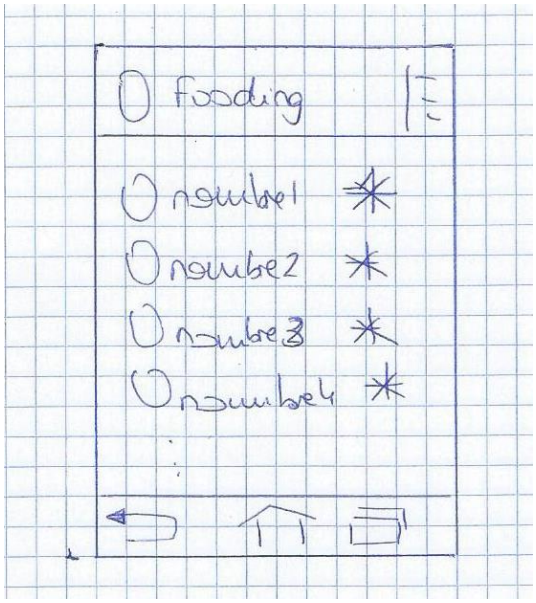
- **Mejora de la aplicación:**

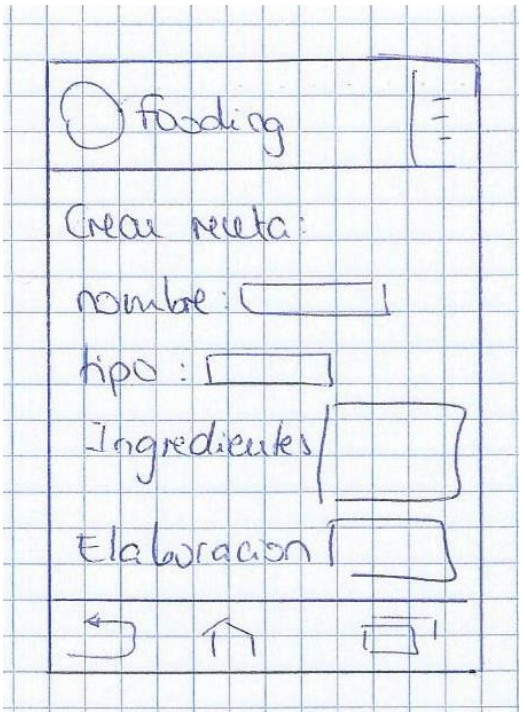
  1. Mejorar la navegación en la aplicación:

     - Cuando un usuario se registra la aplicación le lleva a la pantalla principal, es decir, al listado de recetas.
     - Cuando un usuario se loguea la aplicación le lleva al listado de recetas.
     - Cuando un usuario logueado crea una receta la aplicación le lleva al listado de recetas actualizado con la nueva receta.
     - Cuando un usuario no está logueado no le salen opciones de usuario logueado y viceversa.

A continuación se muestran una serie de prototipos de las nuevas funcionalidades de la aplicación:







2.4 ESTADO DE LA APLICACIÓN

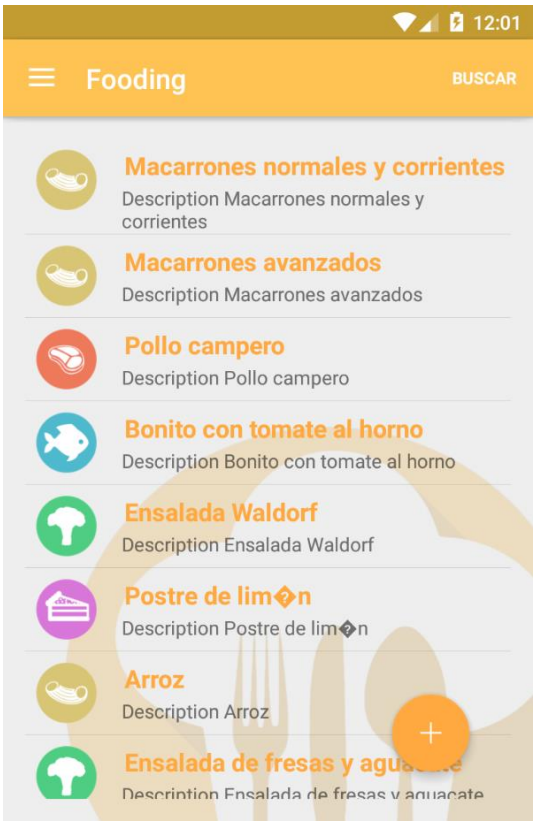
2.4.1 Estado de la aplicación al finalizar el primer sprint

En cuanto al estado de la aplicación, de momento, está desarrollado todo aquello que se propuso el equipo de trabajo para realizar en el primer sprint, las once entradas de la pila ya están implementadas con su correspondiente funcionalidad tanto en la GUI de la aplicación como en el servidor y base de datos.

Todas aquellas funcionalidades cumplen la definición de hecho propuesta en la siguiente sección del documento, proceso.

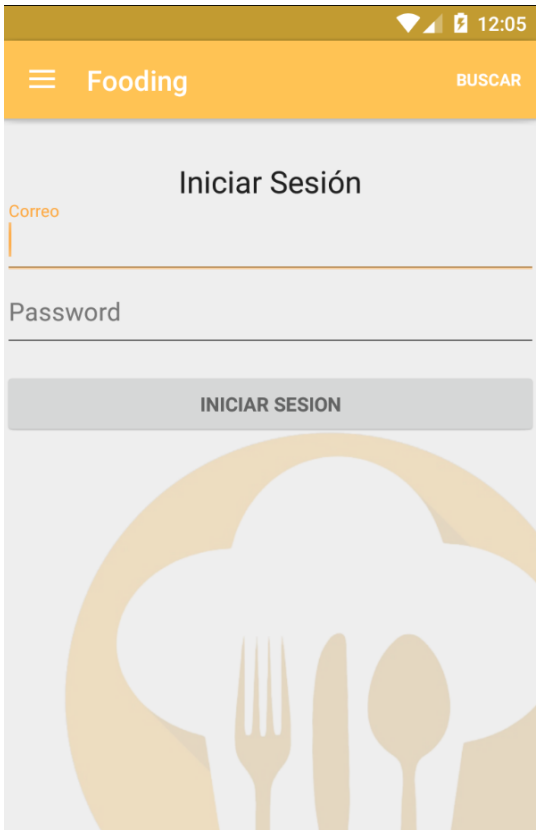
A continuación, se muestran una serie de imágenes con la GUI actual de la aplicación y las funcionalidades desarrolladas.

En cuanto a problemas conocidos se puede destacar la codificación de los caracteres por parte de la base de datos, ya que no reconoce bien las tildes y caracteres especiales, por ello, para el siguiente sprint se ha propuesto solucionar este problema.

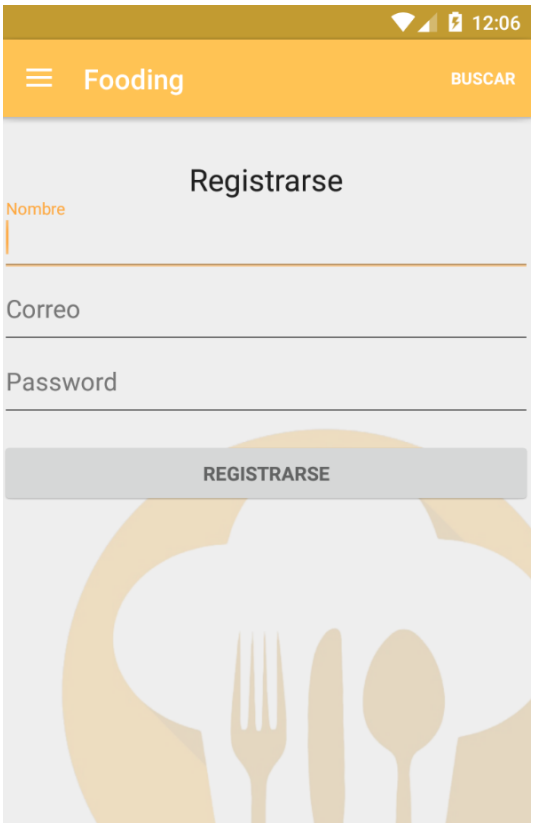


Listado inicial de recetas

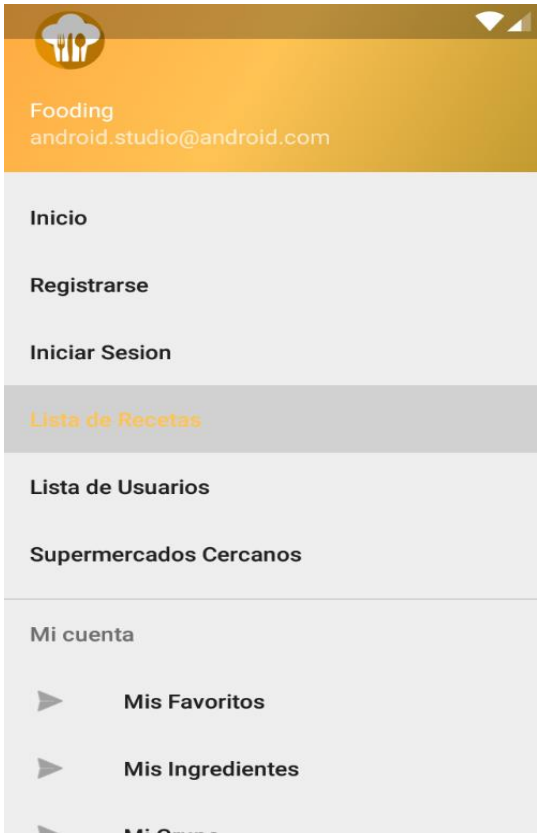
Pop up Búsqueda



Login de usuarios



Registro de usuarios



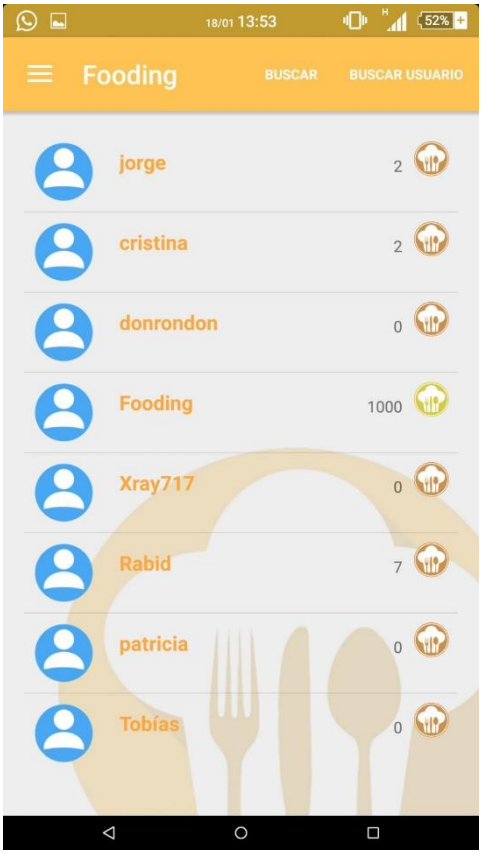
Menú lateral

2.4.2 Estado de la aplicación al finalizar el segundo sprint

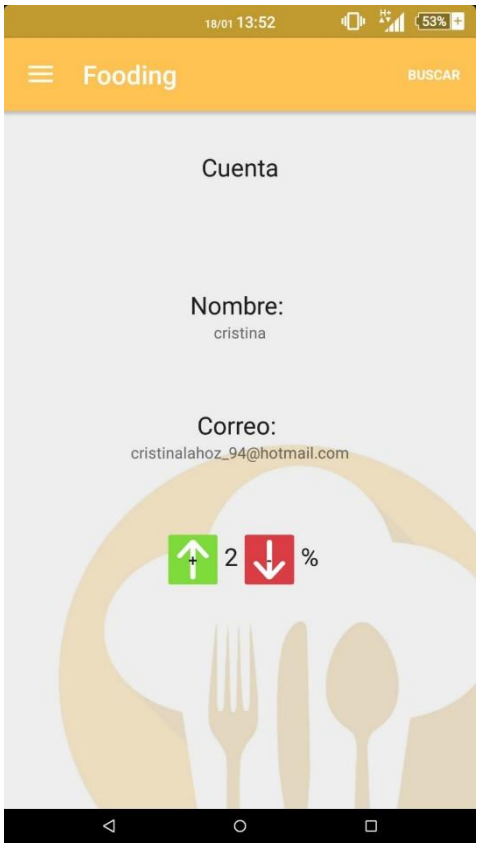
Al término del segundo sprint, de las nueve entradas de la pila para este sprint, se han completado un total de **7 entradas,** aunque cabe destacar que de la entrada de la pila "Sistema de puntuación de usuarios", la única parte no realizada es la de test automáticos en la app, toda la funcionalidad en el servidor y en el cliente **está hecha.**

Todas aquellas funcionalidades cumplen la definición de hecho propuesta en la siguiente sección del documento, proceso.

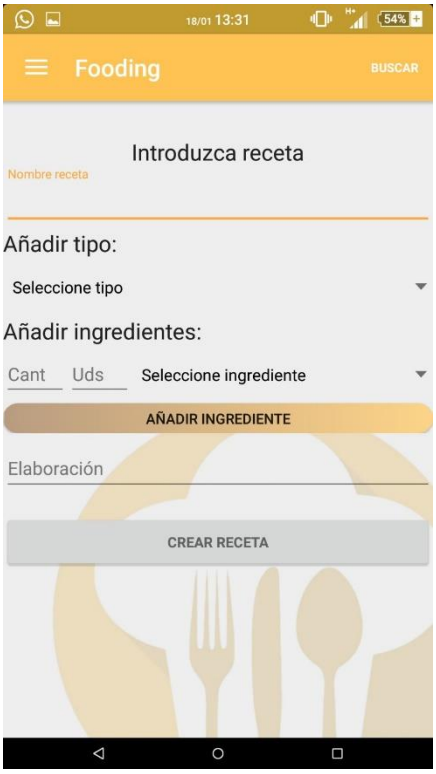
A continuación, se muestran una serie de imágenes con la GUI actual de la aplicación y las funcionalidades desarrolladas.



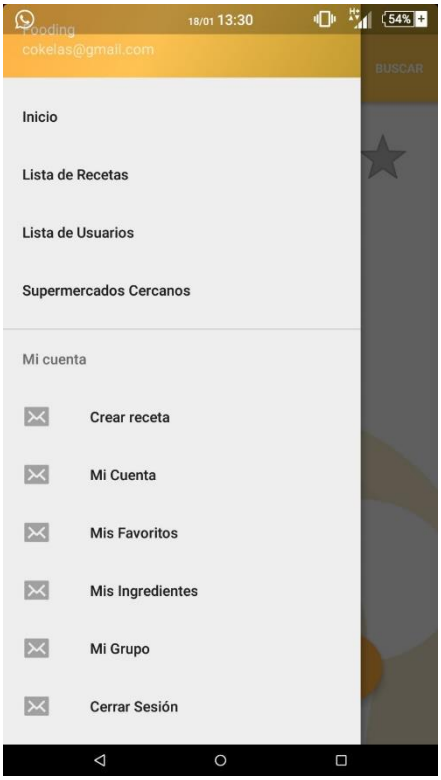
Listado de usuarios



Mi cuenta



Crear recetas

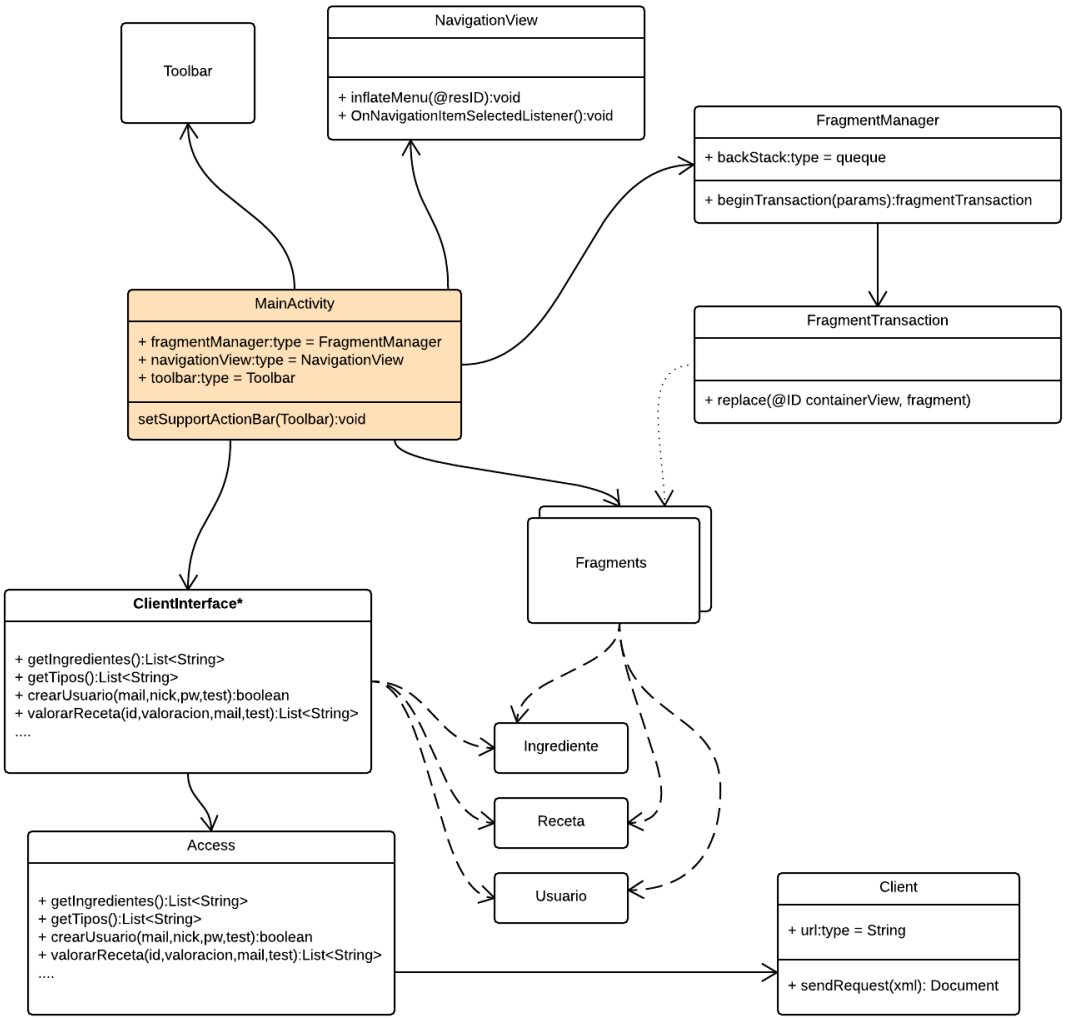


Menú usuario registrado

Listado de recetas con favoritos

2.5 ARQUITECTURA

2.5.1 Aplicación



Tal y como muestra el diagrama, la actividad principal de la aplicación corresponde a la clase MainActivity. Esta clase gestionará toda la arquitectura de la aplicación, formada por 4 partes:

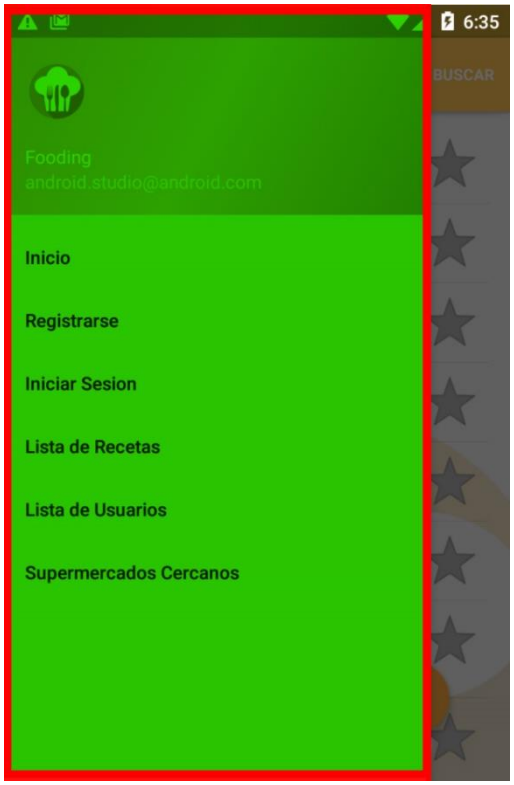
CONTENIDO PRINCIPAL

La aplicación tendrá contenidos dinámicos, y deberá permitir una navegación fluida entre las pantallas, es por eso que se ha decido usar una estructura de Fragments (http://developer.android.com/intl/es/training/appbar/index.html) Esta tecnología permite modificar únicamente una parte de la Actividad principal en la que se encuentre el usuario, de esta forma evitamos volver a cargar elementos estáticos de la aplicación, como es el Menú Lateral y la Barra de Aplicación que se describirán a continuación. A través de la clase

FragmentManager se mantendrá una pila (backStack) almacenando en orden todos los fragments por los que ha navegado el usuario, de esta manera al pulsar el botón atrás, la aplicación volverá al Fragment anterior, y no saldrá de la aplicación, por lo que la experiencia de usuario se verá mejorada considerablemente. Por otro lado, para hacer la transacción entre Fragments se hace uso de la clase FragmentTransaction junto al método replace.

MENU SUPERIOR

La zona superior de la aplicación estará ocupada por una AppBar (http://developer.android.com/intl/es/training/appbar/index.html) Esta barra de acciones estará implementada a través de un objeto Toolbar, que muestre las opciones necesarias. Se ha decidido usar esta implementación ya que así, es muy sencillo modificar las opciones disponibles en esta barra a través del Fragment en el que se encuentre el usuario.



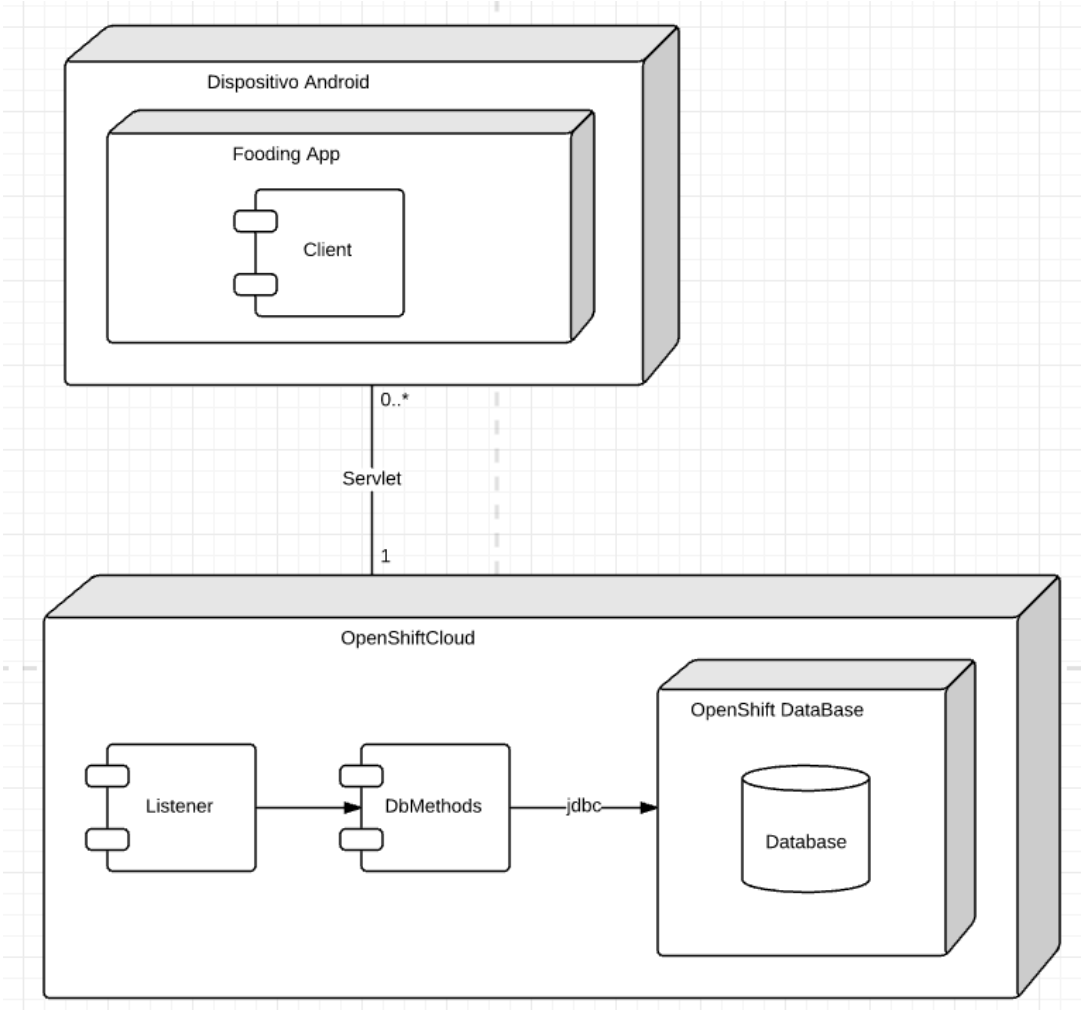
MENÚ LATERAL

Para la implementación del menú lateral se ha usado la clase NavigationView (http://developer.android.com/intl/es/reference/android/support/design/widget/NavigationView.html) Esta clase permite cargar las opciones del menú lateral a través de un .xml con el método (inflateMenu()), el cual puede ser modificado para mostrar distintas opciones según sea necesario, por ejemplo, modificar las opciones según el usuario esté registrado o no. Además, permite instanciar un ActionListener para cada opción del menú.

CONEXIÓN CON LA BD

Para permitir el trabajo concurrente entre el equipo backend junto el frontend, se han separado las clases necesarias para realizar la conexión, asi como las clases que contienen los métodos que realizan las conexión en un paquete aparte con nombre "connection", de esta forma se divide el código evitando mezclas entre el código perteneciente a la GUI y la BD.LA aplicación usará únicamente los métodos de la clase ClientInterface. Dicha clase ejecutará los métodos de la clase Access, que hará uso de la clase Client para enviar y recibir peticiones al servidor.

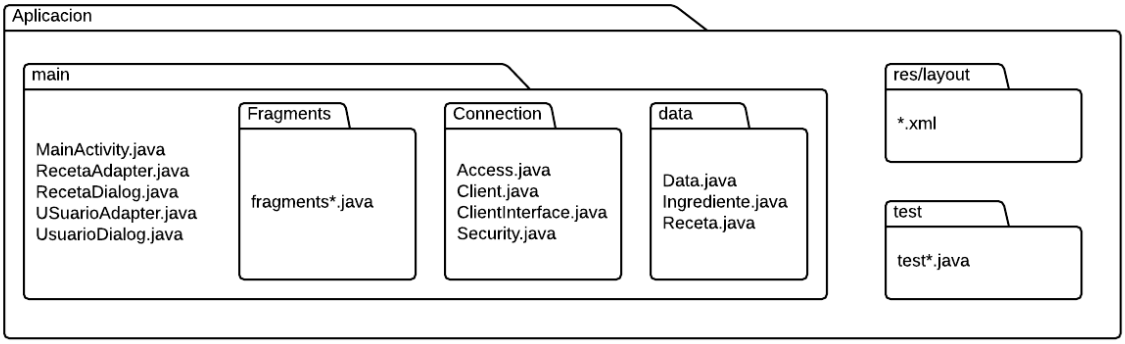
2.5.2 Diagrama de despliegue

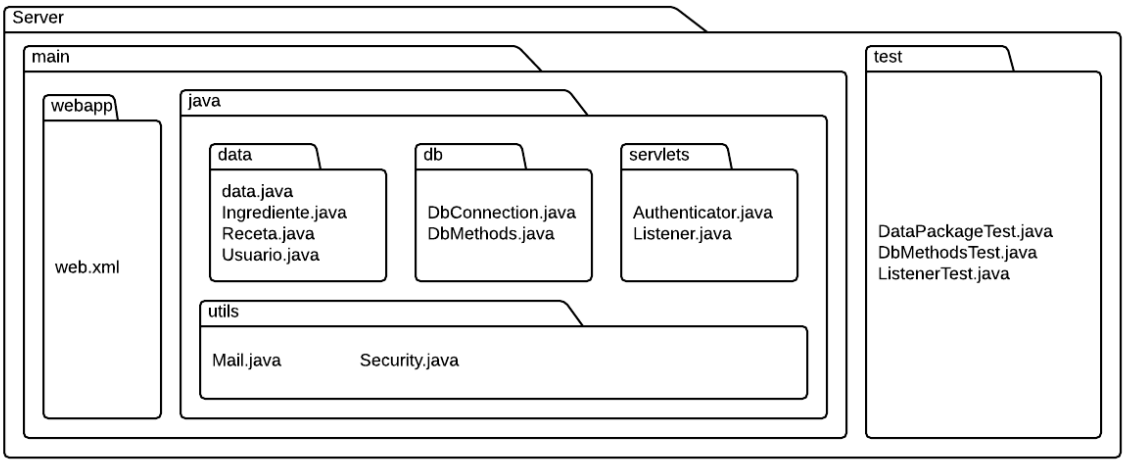


El sistema se compone de dos nodos. El cliente que ejecutará la aplicación en su disposivo Android, y por otro lado el servidor en OpenShift. Este servidor en la nube tendrá un componente Listener, que esté escuchando peticiones de todos los clientes a través de servlets. Se ha decido usar esta comunicación ya que la aplicación no requiere muchas peticiones al servidor, y no se prevee una gran cantidad de usuarios concurrentes. En todo caso, si llegara a dar problemas se podrían replicar los listener y por medio de una cola y redirigiendo a las peticiones según vayan siendo gestionadas.

Por último la base de datos, que ha sido alojada junto al servidor en el mismo "Gear" de Openshift. El listener hará uso de los métodos de DbMethods, el cual accederá a la base de datos a través de jdbc.

2.5.3 Diagrama de paquetes





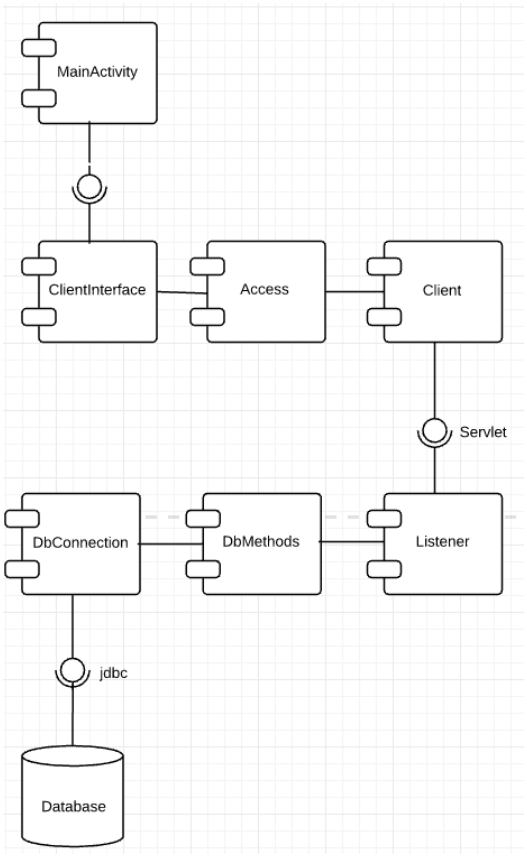
Se ha modificado el esquema de paquetes en la parte del cliente respecto al primer Sprint, ya que la complejidad de la aplicación hacía que el número de clases creciera muy rápidamente y esto dificultaba el desarrollo.

Ahora está organizado en 4 directorios:

- Data: lugar donde almacenar los objetos con los que se va a trabajar
- Connection: clases encargadas de realizar la conexión con la base de datos y enviar/recibir peticiones
- Fragments: clases que extiendan de la Clase Fragment de Android, es decir, cada pantalla
- Raíz: resto de clases, adapters o diálogos, además de la Actividad principal "MainActivity".

Respecto a la parte del servidor, se mantiene la misma estructura, sin cambios.

2.5.4 Diagrama de componentes



A través del diagrama de componentes se ha representado el flujo que recorre la aplicación y grafo que forman sus componentes. El usuario interacciona directamente con el MainActivity, ya sea desde los menús o desde el fragment en cuestión. Este trabajo se lleva a cabo con la interfaz que ofrece la clase ClientInterface.

Esta clase, usa los métodos implementados en Access, que junto al componente "Cliente" realiza una conexión con el Listener del servidor. Esta conexión se realiza a través de servlets. El listener, de manera similar al cliente, hace uso de otros dos componentes, para realizar las consultas a la base de datos a través de jdbc.

3 PROCESO

3.1 DEFINICIÓN DE HECHO

Se considera que una tarea está hecha cuando, después de las fases de análisis, diseño, implementación y pruebas, sea entregada a la dueña del producto y ésta dé su aprobación.

Concretamente, se sigue el siguiente esquema de checklist:

- El código correspondiente a la tarea se encuentra alojado en los repositorios de GitHub.
- La aplicación de Android y la aplicación servidor han pasado todos los tests automáticos después de incorporar el nuevo código.
- En Openshift (el entorno de desarrollo del servidor) debe estar alojado un WAR con la nueva funcionalidad implementada.
- Se ha notificado a la dueña del producto de la finalización de la tarea.
- El código correspondiente a la tarea se encuentra correctamente documentado; si es necesario, se añadirán nuevas entradas a la Wiki del repositorio, incluyendo los diagramas, imagénes, bocetos... que se consideren necesarios.

3.2 TEST Y CONSTRUCCIÓN DEL SOFTWARE

En la aplicación para Android se ha utilizado *Gradle* como herramienta de gestión de dependencias. Para los tests unitarios se ha utilizado la herramienta *Robotium*, pues se ha decidido hacer pruebas de interfaz, probando los métodos y funciones asociadas a las funcionalidades sin llamar directamente a tales métodos.

Para la compilación se ha utilizado también *Gradle*, habiéndose creado un script de compilación que resuelve las dependencias y realiza la compilación de la aplicación, generando un archivo con extensión APK.

No se han utilizado herramientas de cobertura de tests automáticos, pues se ha comprobado al final del sprint el nivel de cobertura de tests manualmente.

En la aplicación servidor se ha utilizado *Maven* como herramienta de gestión de dependencias y compilación. Para los tests unitarios se ha utilizado *Junit* y un

mock de *SpringFramework* (para probar los métodos de acceso desde Internet en un servlet, como GET y POST).

La política seguida en el servidor para los tests fue, implementada la funcionalidad, realizar tests en todos los niveles, desde llamadas a los métodos de acceso a la Base de Datos hasta mocks para comprobar el parseo de datos.

Además, se han realizado pruebas adicionales desde la aplicación y desde distintos navegadores para comprobar su corrección.

Para la compilación también se ha usado *Maven*. Los tests son pasados automáticamente con cada push al repositorio del servidor mediante el script de Maven, que también se encarga de crear un archivo WAR y desplegarlo en Openshift.

Se ha utilizado *JaCoCo* como herramienta para la cobertura de tests automáticos, habiéndose implementado un script para Windows para descargar al archivo HTML que genera JaCoCo con cada compilación en el servidor. Adicionalmente, esta herramienta desglosa en paquetes la cobertura de código, además de mostrar el porcentaje de código cubierto globalmente.

3.3 CONTROL DE VERSIONES

Se han creado en GitHub un total de tres repositorios: un repositorio para la aplicación, otro repositorio para el servidor y un último repositorio para la documentación.

El repositorio del servidor es una copia exacta del repositorio privado de Openshift en el que se aloja la aplicación servidor: el workflow en este repositorio es *centralizado*, utilizando la rama master como rama principal a la que todos los miembros del equipo suben sus cambios.

> Para mantener el repositorio de GitHub sincronizado con el repositorio privado de Openshift se tuvo que crear un enlace a remoto en cada ordenador, así como subir de forma automática al repositorio de GitHub lo mismo que se sube al repositorio de Openshift.
>
> Se decidió crear un conjunto de scripts para automatizar este proceso, siendo estos usados por todos los miembros del equipo. La documentación asociada a estos scripts se encuentra en la Wiki del repositorio, junto a instrucciones de instalación y setup de Openshift.

El repositorio de la aplicación también tiene un *workflow centralizado* en torno a la rama master; aunque para implementar la comunicación entre cliente y servidor se creó una rama (connection_server) separada que iba uniéndose con la rama principal cuando una nueva característica era añadida y testeada.

Por último, en el repositorio para la documentación también se ha seguido un esquema de *workflow centralizado*, pues los cambios en la documentación no

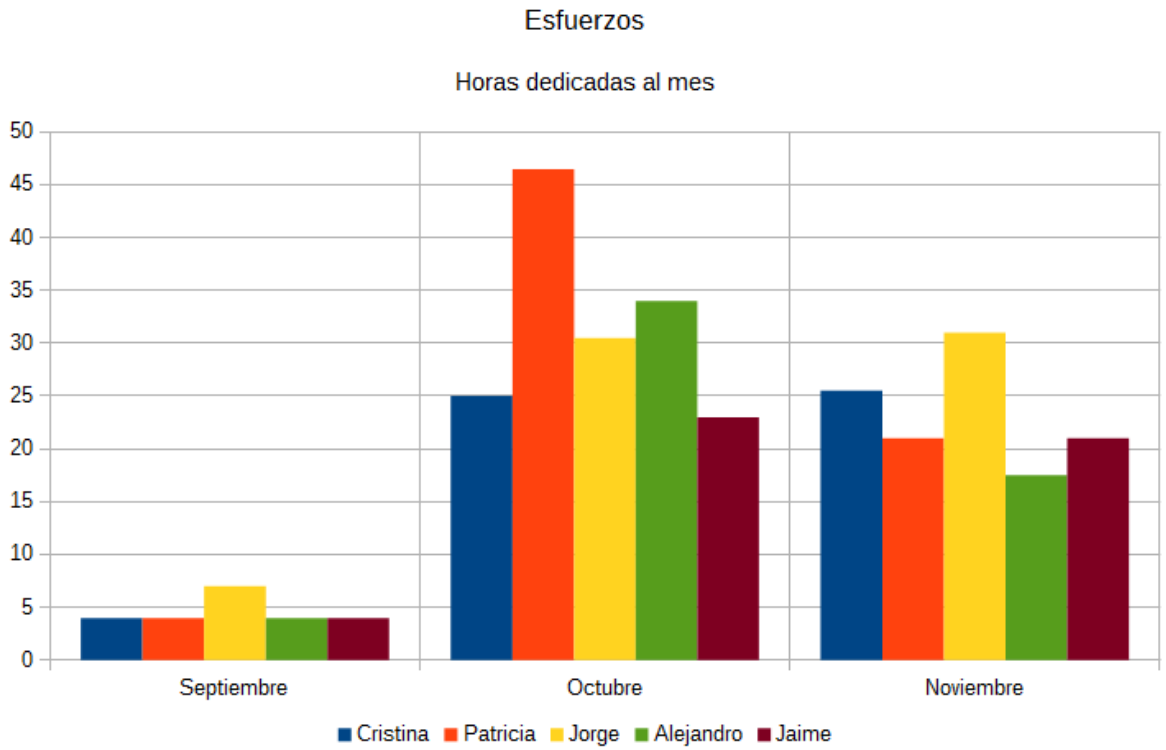
son tan frecuentes como para que supusieran conflictos y no se dio por tanto la necesidad de crear ramas adicionales.

3.4 ESTADÍSTICAS DEL EQUIPO

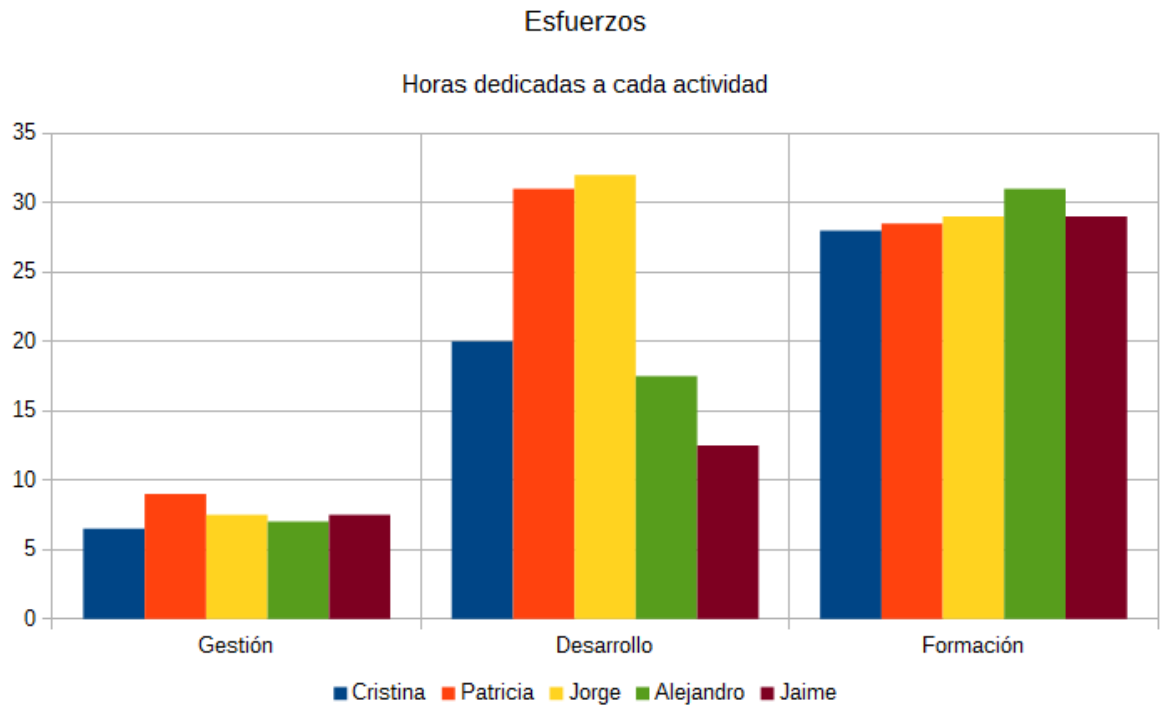
3.4.1 Estadísticas del primer sprint

Esfuerzos

Esfuerzos (medidos en horas) de cada miembro del equipo por mes:



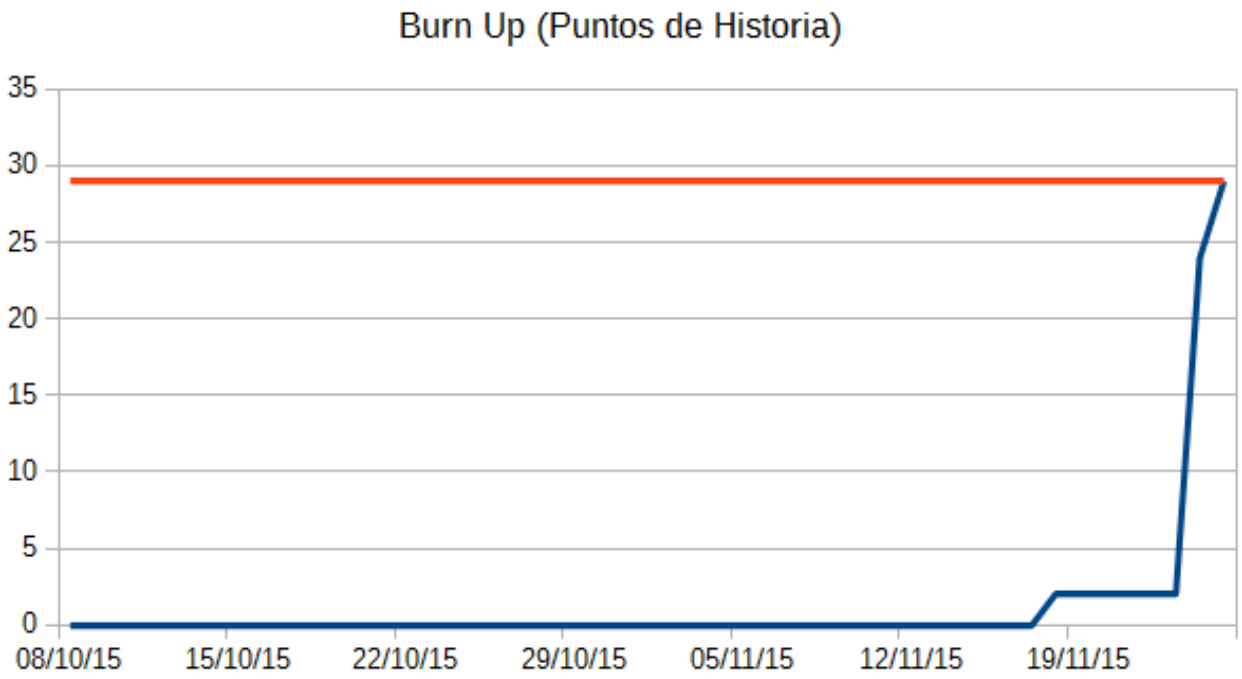
Esfuerzos (medidos en horas) de cada miembro del equipo por actividad:



Velocidad

En este sprint, el equipo ha completado 29 puntos de historia; como no se tienen datos históricos del equipo, no se puede dar un rango de velocidades. La velocidad del equipo será, por tanto, *29 puntos de historia*.

A continuación, se muestra el *diagrama de burn up* o trabajo completado:

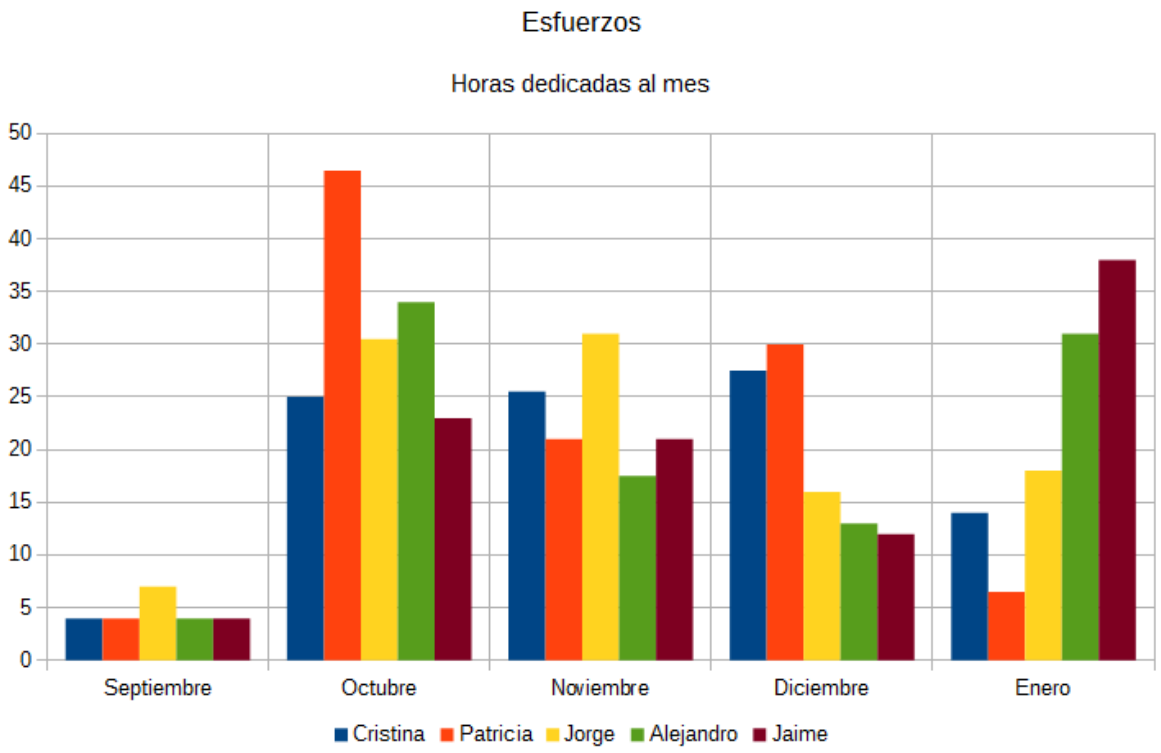


La mayoría de las tareas fueron completadas en el último tramo del sprint debido a problemas que impedían pasar los tests automáticos (problemas de configuración).

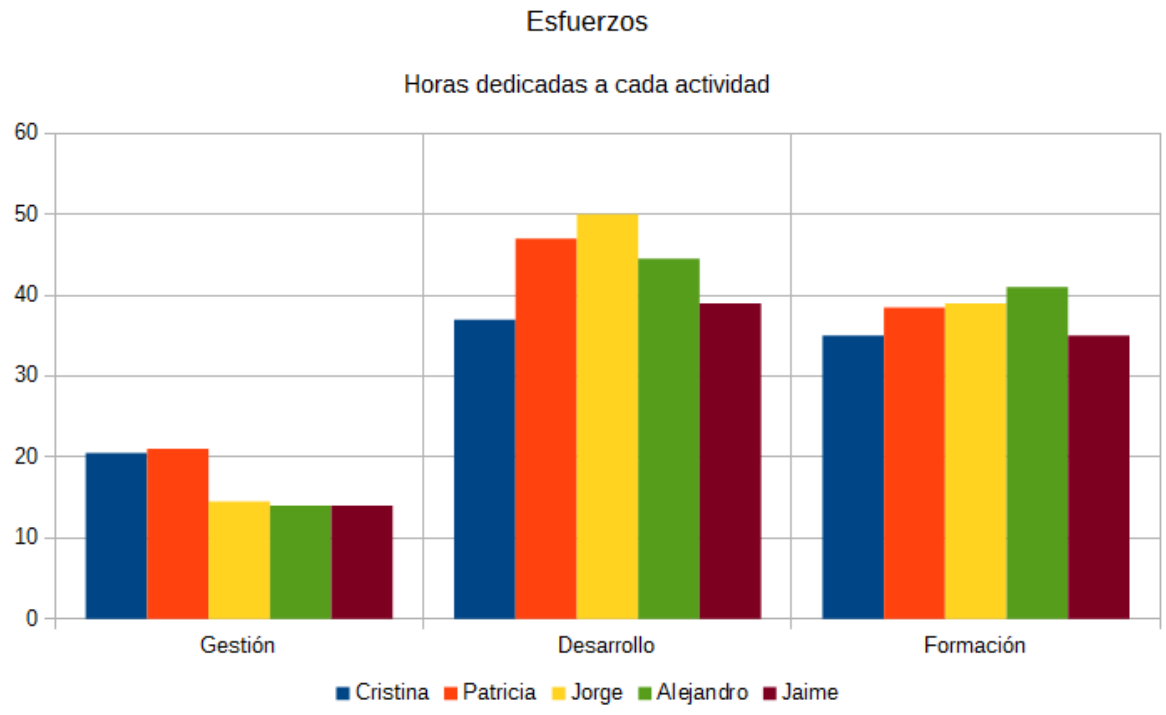
3.4.2 Estadísticas del segundo sprint

Esfuerzos

Esfuerzos (medidos en horas) de cada miembro del equipo por mes:



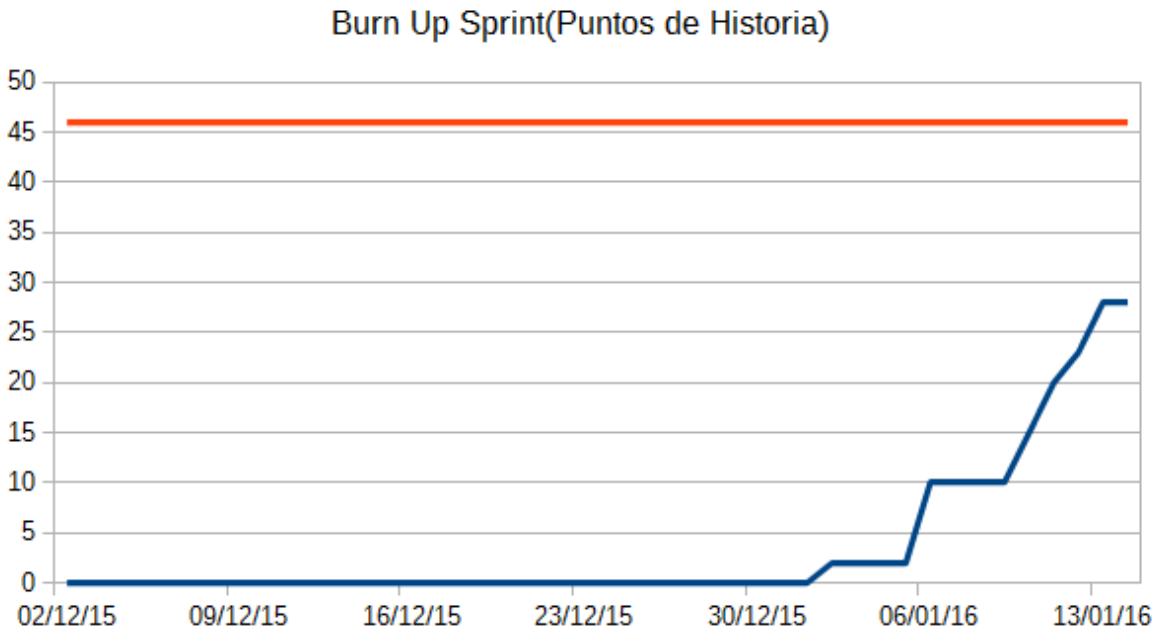
Esfuerzos (medidos en horas) de cada miembro del equipo por actividad:



Velocidad

En este sprint, el equipo ha completado 28 puntos de historia; una cantidad de puntos de historia similar al sprint anterior. Podemos determinar que la velocidad de equipo se encuentra en *28 puntos de historia*.

A continuación, se muestra el *diagrama de burn up* o trabajo completado:



No se han podido completar todos los puntos de historia del sprint.

3.5 RETROSPECTIVA

3.5.1 Retrospectiva del primer sprint

Al finalizar el sprint, se ha llevado a cabo la reunión de retrospectiva del mismo, con el objetivo de analizar los problemas surgidos durante el proceso de desarrollo del producto y de dar soluciones o mejoras respecto a dichos problemas.

En la reunión de retrospectiva se sacaron a la luz los siguientes problemas:

1) Realización de los tests automáticos de la aplicación. No se han podido automatizar algunos tests, en concreto, los de la GUI.

2) Cálculo de la cobertura de código de la aplicación. No se logró configurar para su uso las herramientas para el cálculo de la cobertura de código en la aplicación.

3) Uso ineficiente de la herramienta para la gestión del tablero del proyecto. No se ha utilizado Trello adecuadamente para la visualización del estado en cada instante de tiempo de la aplicación, ni se ha mantenido actualizado.

4) Falta de una correcta comunicación entre los miembros del equipo. Se ha utilizado Whatsapp para comunicarse en el equipo pero ha quedado patente que se trata de una herramienta ineficiente para la gestión del proyecto, comportando pérdidas de información.

A la luz de los problemas expuestos en los puntos anteriores, se ha discutido la prioridad de solución de dichos problemas y se han realizado las siguientes propuestas para remediarlos (por orden de prioridad):

- En relación con el problema 4, se ha establecido la necesidad de realizar reuniones semanales para poner en común aspectos relevantes de cara al desarrollo del proyecto, forzando al menos una reunión cada dos semanas.

- En relación con el problema 3, ha habido un compromiso por parte de todos los miembros del equipo de corregir su actitud y mantener actualizado el tablero de Trello en todo momento durante el desarrollo de la aplicación.

- En relación con los problemas 1 y 2, se ha optado por dedicar horas del segundo sprint a la solución de estos problemas.

3.5.2 Retrospectiva del segundo sprint

Al finalizar el segundo sprint los miembros del equipo se han reunido con el fin de analizar los problemas que han surgido durante este proceso de desarrollo del producto y dar soluciones y mejoras.

En relación con los problemas identificados en el primer sprint, como que la herramienta Trello no era actualizada adecuadamente se ha podido ver en este sprint que los miembros de equipo estaban más concienciados y actualizaban dicha herramienta de manera más habitual.

También, los problemas que tenía el equipo en relación con la falta de comunicación han sido solucionados en este sprint realizando reuniones en las que participaban todos los miembros de equipo para dejar claro los objetivos y clarificar situaciones que podían traer problemas.

En la reunión de retrospectiva se sacaron a la luz los siguientes problemas:

1) Ambigüedad de las entradas de la pila: no hay una definición o descripción clara en Trello, ni bocetos de la interfaz accesibles a todo el equipo.
2) Problemas con el servicio de hosting en la nube Openshift: durante unos días, falló al subir nuevo código al servidor, haciendo difícil continuar el trabajo. A día de hoy todavía no se sabe cuál fue la causa del fallo

A la luz de los problemas expuestos en los puntos anteriores, se han realizado las siguientes propuestas para remediarlos (por orden de prioridad):

1) En relación al problema 2, se reservará tiempo del siguiente sprint para acotar el fallo y analizar otros servicios de hosting en la nube si es necesario migrar el servidor.
2) En relación al problema 1, se ha decidido guardar la descripción de las entradas de la pila en Trello, junto a unos bocetos de la interfaz, que serán accesibles a todos los miembros del equipo.

4 CONCLUSIONES

4.1 SUMARIO

El producto a desarrollar consiste en una aplicación de recetas de comida para smartphones que ayude a estudiantes a hacer la comida.

Inicialmente el equipo se dividió en dos equipos, uno más centrado en la **app,** y otro más centrado en el **servidor,** empleando como medio de comunicación entre ambos "subequipos" **Whatsapp**, y usando Trello como tablero central del proyecto.

Una vez establecida la organización del equipo, se abrieron 3 repositorios para el control de versiones de los ficheros de la aplicación: un repositorio para la documentación, otro para el código de la aplicación móvil y otro para el código de la aplicación del servidor.

Asimismo, durante el desarrollo del proyecto se fueron diseñando en la medida de lo posible tests automáticos que comprobaban el correcto funcionamiento de la aplicación cada vez que se realizaba algún cambio en los repositorios, así como permitir al equipo medir la cobertura del código.

Al finalizar el primer sprint, a pesar de haber cumplido los objetivos que nos propusimos para el mismo, surgieron algunos problemas en relación con la gestión y la comunicación del equipo, pero se forzaron a mejorar y corregir para el segundo sprint.

En el segundo sprint se mejoró la comunicación en el equipo, realizándose reuniones periódicas, y se concienció a todos los miembros del equipo para que utilizaran Trello de forma adecuada.

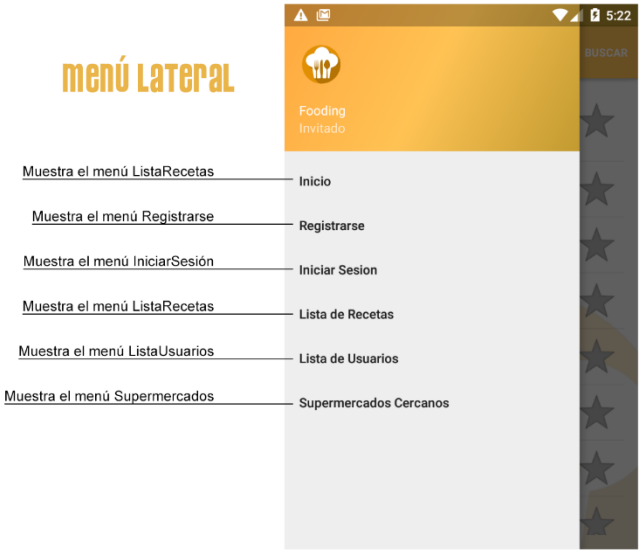
Asimismo se detallaron más las condiciones de satisfacción de las entradas de la pila del sprint, pero no se pudieron completar todas. Esto se debe a que, aunque las horas estimadas para completar las entradas de la pila eran las mismas, los puntos de historia estimados eran superiores; por tanto, podemos concluir que el equipo realizó una mala estimación de horas.

4.2 CUMPLIMIENTO DE OBJETIVOS

- El código del proyecto se aloja en GitHub y se trabaja de forma habitual contra git: **sí** (mirar los commits y actividad de los integrantes del equipo en los repositorios de GitHub).
- Cobertura de tests automáticos: **81%** en la aplicación servidor y **47,52%** en la aplicación de Android. Por la parte del servidor, se ha calculado la cobertura del código con la herramienta **JaCoCo** y por la parte del cliente, el cálculo se ha hecho a mano:
    - El cliente posee **1515** líneas en total
    - De las cuales **720** están cubiertas por los test.
    - 720 líneas de 1515 corresponden al 47,52% de código cubierto.
- Historias de usuario/requisitos implementados a lo largo del proyecto evaluados con buenos criterios de aceptación:
    - Durante el primer sprint: **100%.** Todas las entradas de la pila realizadas han sido validadas por la dueña del producto siguiendo las condiciones de satisfacción planteadas
    - Durante el segundo sprint: **77,77%.** De las 9 entradas de la pila planteadas, 7 fueron validadas por la dueña del producto atendiendo a las condiciones de satisfacción.
    - Globalmente: **90%**. De las 20 entradas de la pila inicial, 11 de la pila del primer sprint y 9 de la pila del segundo sprint, 18 fueron validadas por la dueña del producto atendiendo a las condiciones de satisfacción.
- Definición de hecho clara, usada e incluye que se pasen todos los tests automáticos: **sí**.
- Documentación arquitectural (vista de módulos/componente y de despliegue, discusión sobre razones arquitecturales): **sí**.
- Cumplimiento suficiente de requisitos/características: **no**. Para este sprint no se han realizado todas las entradas de la pila planificadas al comienzo del mismo.
- Control de esfuerzos: **sí** (se sigue el control de esfuerzos propuesto).
- Compilación y gestión de dependencias basada en scripts (despliegue automático, generación de binarios, triggers automáticos...): **sí** (la aplicación servidor incluye despliegue automático, tests automáticos, generación de binarios automática; la aplicación de Android incluye generación de binarios automática).
- Toda la documentación del proyecto está bajo control de versiones: **sí** (la documentación se encuentra en GitHub).

- Tests de aceptación automáticos para criterios de aceptación implementados en a lo largo del proyecto:
    - Durante el primer sprint: **6** (búsqueda por nombre, búsqueda por ingredientes, búsqueda por tipo, registro de usuarios, login de usuarios, logout de usuarios).
    - Durante el segundo sprint: **7** (modificar comensales de una receta, listado de usuarios, crear y publicar recetas, valorar receta, visualizar valoración media de una receta, visualizar pantalla de usuario, sistema de favoritos)

5 ANEXO: MANUAL DE USUARIO



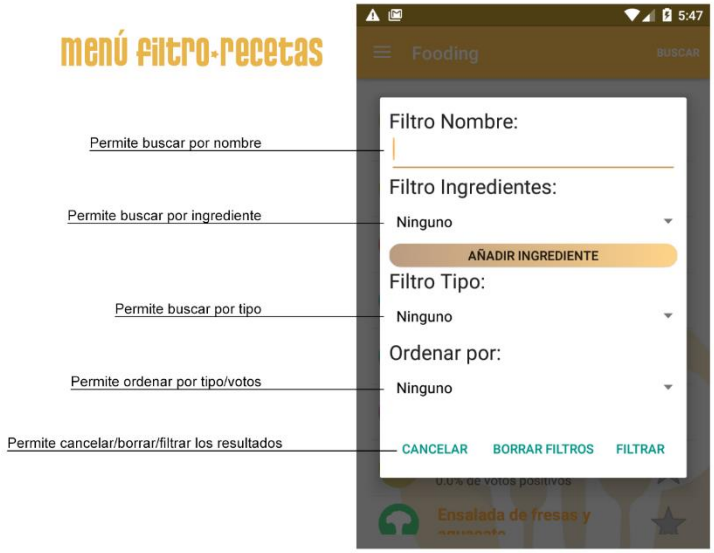
La primera vez que un usuario abre la aplicación puede ver el menú lateral que se muestra en la imagen superior, se puede ver que un usuario tiene la opción de ver el listado de recetas que hay en la aplicación, además de un listado de usuarios que están registrados en la aplicación.

Tiene también la opción de registrarse, o si ya está registrado puede iniciar sesión.

Si un usuario pulsa en *Lista de recetas* se le mostrará un listado con todas las recetas disponibles en la aplicación, este listado tenemos opciones como buscar una receta añadiendo los filtros que se consideren, también, se muestra el tipo de la receta, el nombre, el porcentaje de votos positivos que ha recibido dicha receta, además, una

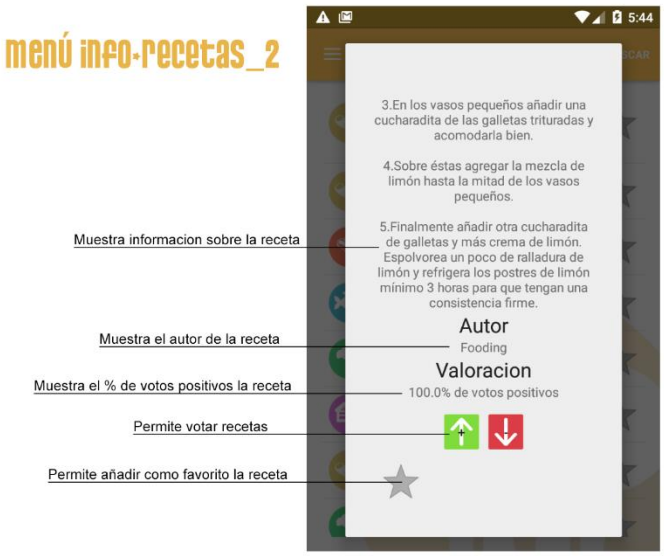
estrella a la derecha de cada receta permitirá a los usuarios marcar una receta como favorita. Esta opción solo está disponible para usuarios que hayan iniciado sesión.

En esta pantalla también tenemos la posibilidad de ver la información de una receta pulsando en ella.



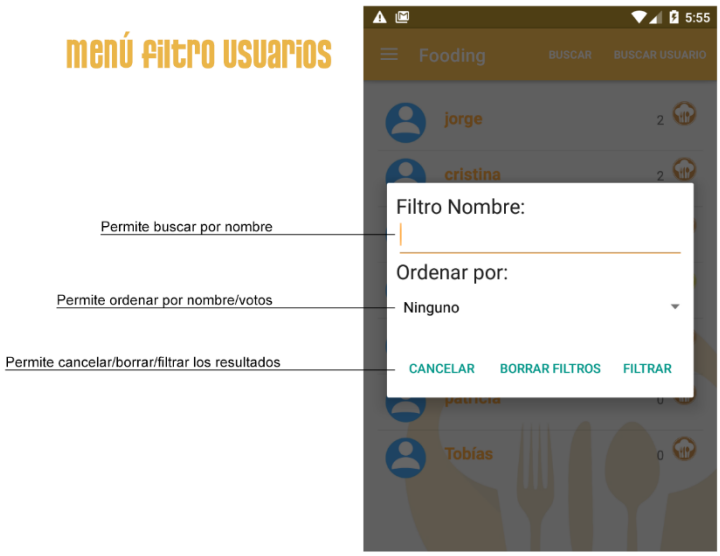
La opción nombrada anteriormente para buscar recetas se muestra en la imagen superior, permitiendo buscar recetas por nombre, por ingrediente, pudiendo añadir hasta tres ingredientes, por tipo de la receta (carne, pescado, pasta, postre y verdura). También permite la opción de ordenar por el tipo de la receta o por el porcentaje de votos positivos.

Para concluir con esta funcionalidad se pulsa en una de las opciones, *filtrar*, *cancelar* o *borrar filtros*.

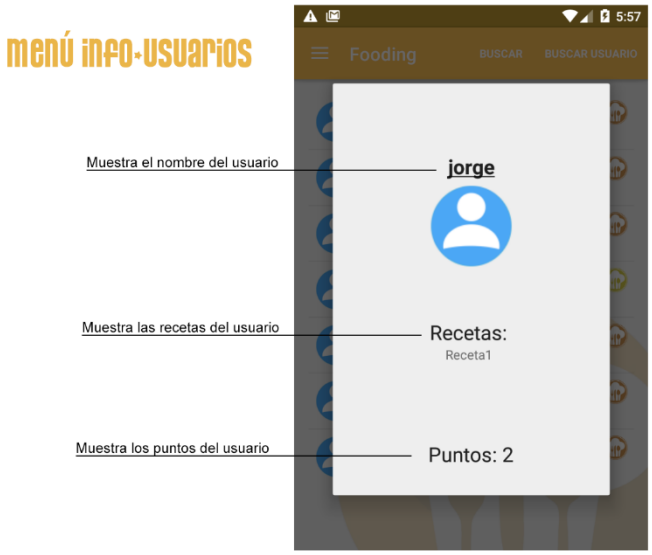


La información de una receta la podemos ver reflejada en las dos imágenes superiores. Esta información está formada por el nombre, el tipo, los ingredientes, la elaboración, una opción para modificar el número de comensales, el autor de la receta, la valoración, y las opciones de votar una receta y añadir una receta a favoritos que solo están disponibles para usuarios que hayan iniciado sesión.

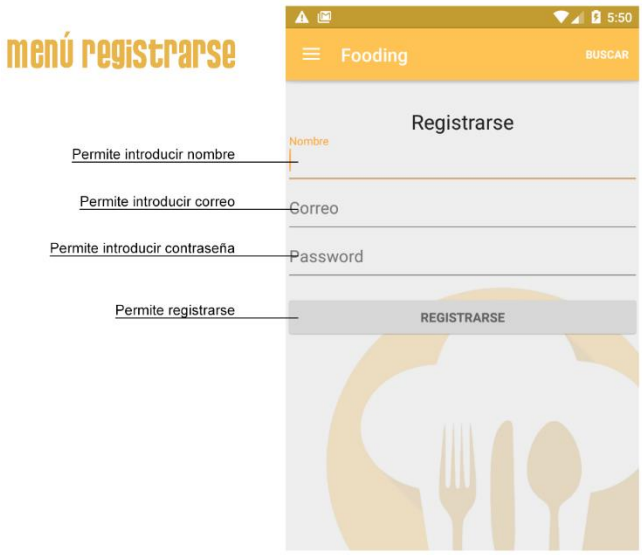
Por otro lado tenemos el listado de usuarios, en él se permite buscar usuarios y en este listado se muestra el nombre de usuario, junto con la puntuación de dicho usuario, también, está disponible la opción de ver la información de un usuario.



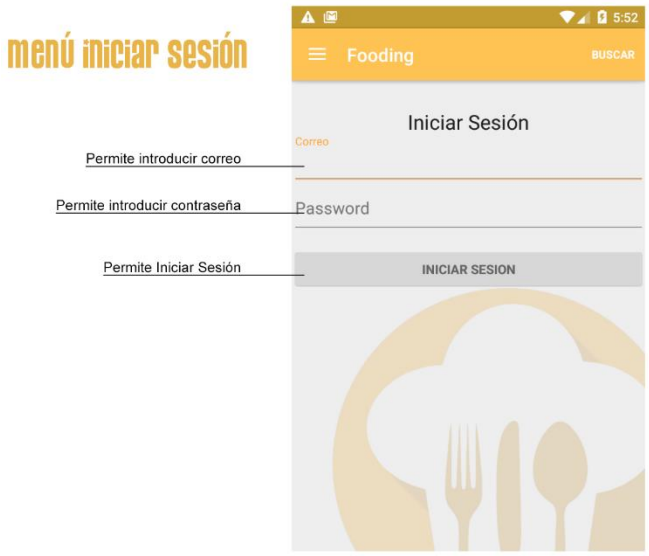
La opción de buscar usuarios permite realizar la búsqueda por nombre de usuario y ordenar el listado de usuarios alfabéticamente por nombre o por el número de votos, para finalizar esta acción es necesario pulsar en *cancelar*, *borrar filtros* o *filtrar*.



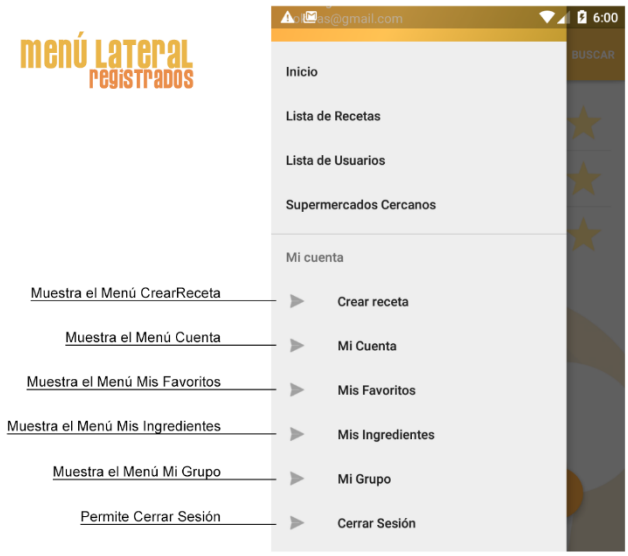
En la información de un usuario podemos ver el nombre del usuario que queremos, las recetas que ha creado este usuario y el número de puntos que ha obtenido ese usuario.



Cuando un usuario decide registrarse tiene que rellenar el formulario de la imagen anterior, introduciendo un nombre de usuario, un correo electrónico y la contraseña. Para que el registro sea completo deberá pulsar en *registrarse* y automáticamente se enviará un correo electrónico a la dirección indica con un link para validar la cuenta, en caso de no validar la cuenta el registro no será completo.



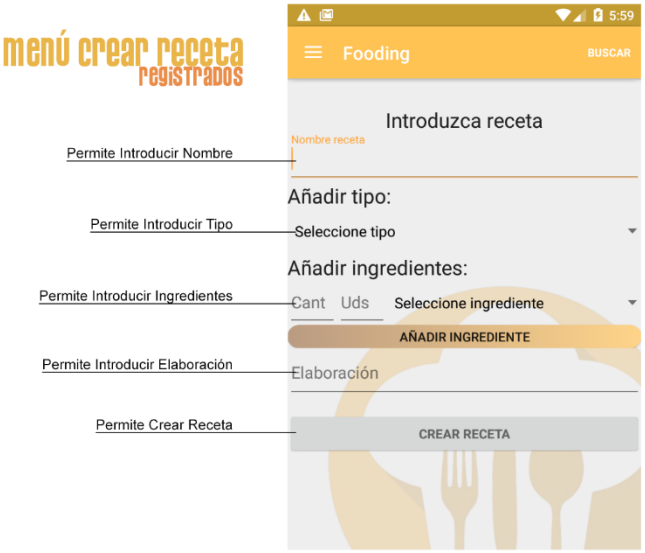
Una vez el usuario se ha registrado, para iniciar sesión tiene que pulsar en el menú de inicio en *iniciar sesión* y rellenar los campos con el correo que había introducido en el registro, la contraseña correspondiente y pulsar en *iniciar sesión* si todo ha sido correcto el usuario será redirigido al listado de recetas, modificándose también el menú de inicio.



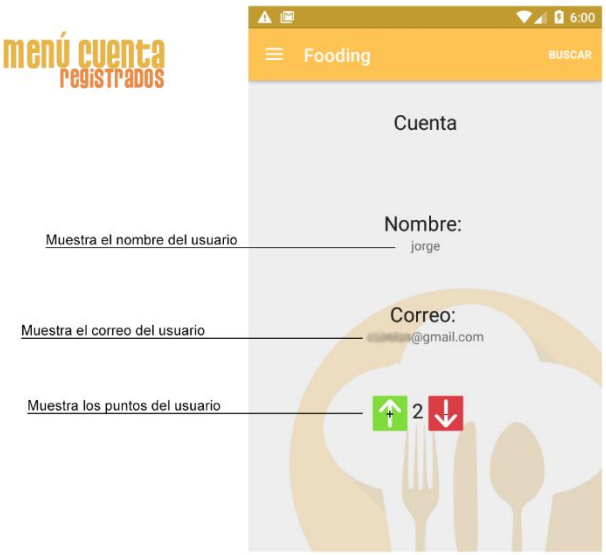
El menú anterior solo está disponible para usuarios que hayan iniciado sesión, tiene más funcionalidades que el resto de usuarios.

Estos usuarios tienen la opción de crear una receta, de ver las características de su perfil, ver las recetas que tienen guardadas como favoritas, los ingredientes que ha añadido un usuario y el grupo de usuarios al que pertenece. Estas dos últimas funcionalidades todavía no están implementadas.

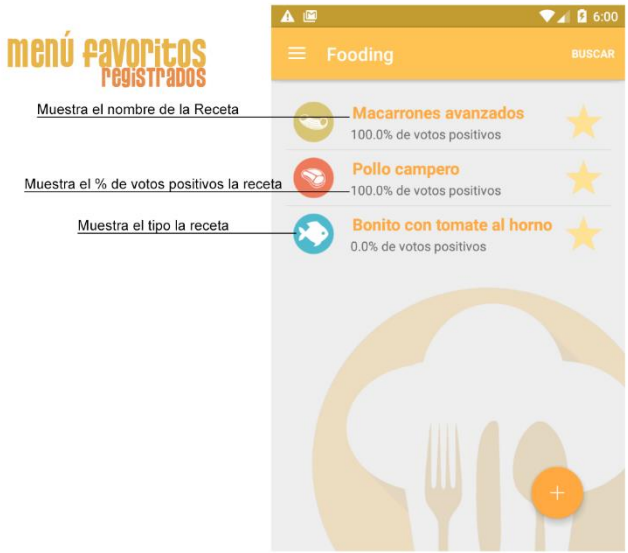
También pueden cerrar sesión.



Un usuario registrado para crear una receta debe pulsar en el menú lateral en *crear receta* y rellenar los campos del formulario anterior de acorde con la receta que desee añadir, una vez completados todos los campos, es importante rellenar todos porque en caso contrario la receta no será creada el usuario pulsará en *crear receta*, si la receta ha sido creada correctamente el usuario será redirigido al listado de recetas donde podrá ver la receta que ha añadido.



Un usuario que haya iniciado sesión podrá ver también la información de su cuenta, el nombre de usuario, el correo electrónico y los puntos que tiene.



Cuando un usuario pulsa en el menú lateral en *mis favoritos* se le muestra un listado con las recetas que ha marcado como favoritas del listado de recetas. En este listado se muestra el nombre de las recetas, el porcentaje de votos positivos y el tipo de la receta.